Assessment of Applying the PMaC Prediction Framework to NERSC-5 SSP Benchmarks

Summer 2006

Author: Noel Keen

Introduction

NERSC procurement depends on application benchmarks, in particular the NERSC SSP. Machine vendors are asked to run SSP benchmarks at various scales to enable NERSC to assess system performance. However, it is often the case that the vendor cannot run the benchmarks at large concurrency as it is impractical to have that much hardware available. Additionally, there may be difficulties in porting the benchmarks to the hardware.

The Performance Modeling and Characterization Lab (PMaC) at San Diego Supercomputing Center (SDSC) have developed a framework to predict the performance of codes on large parallel machines. The goal of my work for NERSC was to apply the PMaC prediction framework to the NERSC-5 SSP benchmark applications and ultimately consider the accuracy of the predictions. Other tasks included identifying assumptions and simplifications in the process, determining the ease of use, and measuring the resources required to obtain predictions.

The primary goal of the PMaC framework is to assess machine hardware for procurement. Using the PMaC prediction framework, we wish to predict full-scale benchmark performance on target machines using available hardware. First, tracing data is collected from full-scale runs on existing Alpha processor based machines. This data is then combined with raw system measurements of the target machine using memory and network bandwidth benchmark kernels, which do not require a large machine. Finally, the PMaC framework is used as a machine simulator to make predictions about application performance on target machines. This simulator could also be used to change certain machine parameters to answer “what-if” questions for alternative machine configurations.

PMaC Framework Overview

Currently, vendors port benchmarks and full application codes to new machines and report results. Large applications can be difficult to port and vendors may not have a large-scale machine ready for benchmarking. Instead, the PMaC framework would only require that vendors run small kernel benchmarks (to measure memory and network bandwidth) on the target machines and do larger simulation calculations on other available machines. The collected data could also be useful for identifying gross bottlenecks such as network latency, L1 cache misses, and main memory bandwidth saturation. The profiling information is at the basic block level.

The PMaC framework is a collection of tools in C, Perl, Python, and shell scripts. It is also a process that must be followed carefully. The framework is automated in the sense that it requires only minor application code modifications, however there are many steps that must be completed and these are not currently automated. The framework is capable of collecting detailed serial and parallel communication tracing information about the application codes. A commercial software package maintained by CEPBA (Centro Europeo de Paralelismo de Barcelona) is required to use the PMaC framework. In particular the mpidtrace library is used to collect detailed information from every MPI message passed through the network during application execution. Dimemas is a network simulator from CEPBA that is used by the PMaC framework to make the final predictions.

Target Machines

The machines of which a prediction is desired must be determined. The target machines for this study were:

1) bassi.nersc.gov (P5)
2) IBM POWER5+ system
3) Seaborg.nersc.gov (P3+)
4) Jacquard.nersc.gov (Opteron)
5) Dual-Opteron system
6) Davinci.nersc.gov (Itanium2)
7) SGI system

NERSC-5 SSP Benchmarks

The applications which will be used to benchmark the target machines must be determined, as well as the concurrency.

NERSC-5 SSP Benchmarks

Benchmark	**Size**	**num procs**
cam31	**Medium**	**56**
	Large	**240**
Milc	**Small**	**4**
	Medium	**64**
	Large	**256**
Gtc	**Small**	**4**
	Medium	**64**
	Large	**256**
Paratec	**Small**	**4**
	Medium	**64**

	Large	**256**
madbench	**Small**	**4**
	Medium	**64**
	Large	**256**
Pmemd	**Small**	**4**
	Medium	**64**
	Large	**256**

The gamess application was not considered in this work because it uses LAPI, which does not work with mpidtrace (CEPBA are working on a version of tools to support LAPI). Gamess can use MPI, but the performance is poor. (http://www.nersc.gov/projects/ssp.php/)

PMaC Framework Outline

An outline of the PMaC framework steps:

1) Obtain cache system data for all target prediction machines
2) Memory operations collection. Involves the metasim tracer (semi-cycle accurate simulator) collects memory references, floating-point operations
3) Network performance data collection. CEPBA tools collect detailed data for each MPI call.
4) Measure latency and bandwidths of machine on a few CPU's.
    a) MAPS – Memory Access Pattern Signature – memory bandwidth
    b) netbench -- measure network bandwidth and latency
5) Convolving and parameter sensitivity studies

Memory operations collection

The PMaC framework requires all target applications to run full-scale on a machine with Alpha processors. Therefore, the NERSC-5 applications were ported and run on Lemieux at Pittsburg Supercomputing Center, which is a 750-node Compaq ES45 with 4-way 1-GHZ ev68 processors. The framework makes use of the ATOM toolkit which contains a binary rewriter and is only available on Alpha processors. The PMaC team is developing a binary rewriter for IBM processors.

The framework breaks up the data collection into serial tracing and communication tracing. The serial tracing still requires the application to be run at the various scales of interest, but the data collected is only memory references and floating point instructions, ignoring network communication. This serial tracing is what must be done on the Alpha processors of Lemieux.

There are 3 calls that must be added to the source of each application which allow the framework to know exactly the beginning and end of the code to be analyzed as well as the process ID. Each application executable must be instrumented using ATOM, which produces another executable. ATOM enables profiling at the basic block level. Each application at each size is then run twice (pass 1 and pass 2) with the binary-instrumented executable. The data is collected on a per-processor basis, i.e. a 64-way run would output 64 trace data files. The first pass is to obtain .ijbb files which contain fundamental information for each basic block, sorted by the number of times the block is executed. Then each .ijbb file is processed (using ATOM) to obtain .jbb and .badbb files (again, for each processor of a given run). The .badbb files contain a list of the basic blocks of the code that issued a large amount of memory references (above some threshold). These basic blocks will be analyzed further. The full application is then run again using a new binary-instrumented executable. This is the step in which the target machine cache information is used; it is built into the executable and will simulate each cache system. The result is a .imt file for every processor which is post-processed to obtain a .systemXX file, where XX is the target machine ID. For this work, there were 6 systems, so there were 6 output files for each processor, for each application and size combination. Note that there are actually 7 target machines, but because the two Opteron machines have the same cache structure (as specified to the framework), this system only needed to be traced once. The differences between these two Opteron targets would be contained in the MAPS measurements described below.

For the serial tracing, the PMaC framework represents each target machine as a very simplified cache system. To illustrate, consider this example cache structure data that represents the machine bassi.nersc.gov, which has IBM P5 cores:

- L1 buffer size: 32 KB
- L1 associativity: 4 way
- L1 line size: 128 B
- L2 buffer size: 960 KB
- L2 associativity: 10 way
- L2 line size: 128 B
- L3 buffer size: 36 MB
- L3 associativity: 12 way
- L3 line size: 256 B

There are some important assumptions regarding this simplified cache system representation. For the cache buffer and cache line sizes, only floating-point data is considered, i.e. instruction and integer data is ignored. The buffer sizes are per core and the effects of sharing are ignored. A node with an L3 buffer size of 400MB shared among 4 CPU's would be simulated as 4 CPU's each with an independent L3 buffer size of 100MB. There is also a flag named "random_replacement" that can be true or false. Instructions from the PMaC team were to set this flag to false for all machines except the IBM POWER3. One difficulty in obtaining this data was that the NERSC-5 vendor data was incorrect regarding the details of the cache system in two places.

The MetaSim tracer is a semi-cycle accurate simulator that collects data for each basic block, utilizing statistical sampling. The data collected includes the number of executions, number of memory references, number and type of Floating-Point operations, a stride analysis, cache hit rate for each level of cache, branch intensity, number of TLB's. For each memory references, the tracer determines if it was a stride one access, stride two access, up to stride 64. Strides above 64 were considered random. In determining the stride values, the tracer stores the last 2048 addresses. Events such as TLB misses and memory prefetch are counted, but are not currently used in the simulation.

Communication operations collection

Seaborg was chosen to collect the parallel communication tracing data using the CEPBA tools. To use the CEPBA tools, each hostid requires a license. The first step is to link each application with the mpidtrace library and run full-scale. Linking with this tracing library should not change the behavior of the code, however the cam benchmark failed to execute properly. The exact location of the failure was located and reported to CEPBA. The issue was not resolved. Data is collected for every MPI call, therefore the tracing overhead (run-time and data storage) will be largest for those applications with many MPI calls. The next step is to use the tool mpi2trf to post-process all of the data files (one for each processor) into a single trace file (note: for some application runs, this was a large file). The final step would be to use Dimemas. This work does not include any Dimemas simulation as this is part of the Convolving step that the PMaC team was working on. Inputs to Dimemas include on-node and off-node bandwidth values, as well as on-node and off-node latency values. Note that one value is used to represent each of these measurements. There is also a parameter "number of buses" which was "determined by experience"; for Seaborg the value was 6.

Disk I/O modelling

To model application disk I/O, the mpidtrace library requires a system library on IBM systems named DPCL. On both Seaborg and Bassi, DPCL was not working and problem tickets were sent to IBM. Therefore a static version of mpidtrace was obtained from CEPBA to use on Seaborg and we simply ignored I/O. PMaC is working on better ways to model I/O. The madbench application, largely an I/O benchmark, would not be expected to yield accurate prediction results because of the large percentage of time spent in I/O.

Computing resources required by PMaC framework

The computing resources required to collect tracing data was substantial (See Figure 1).

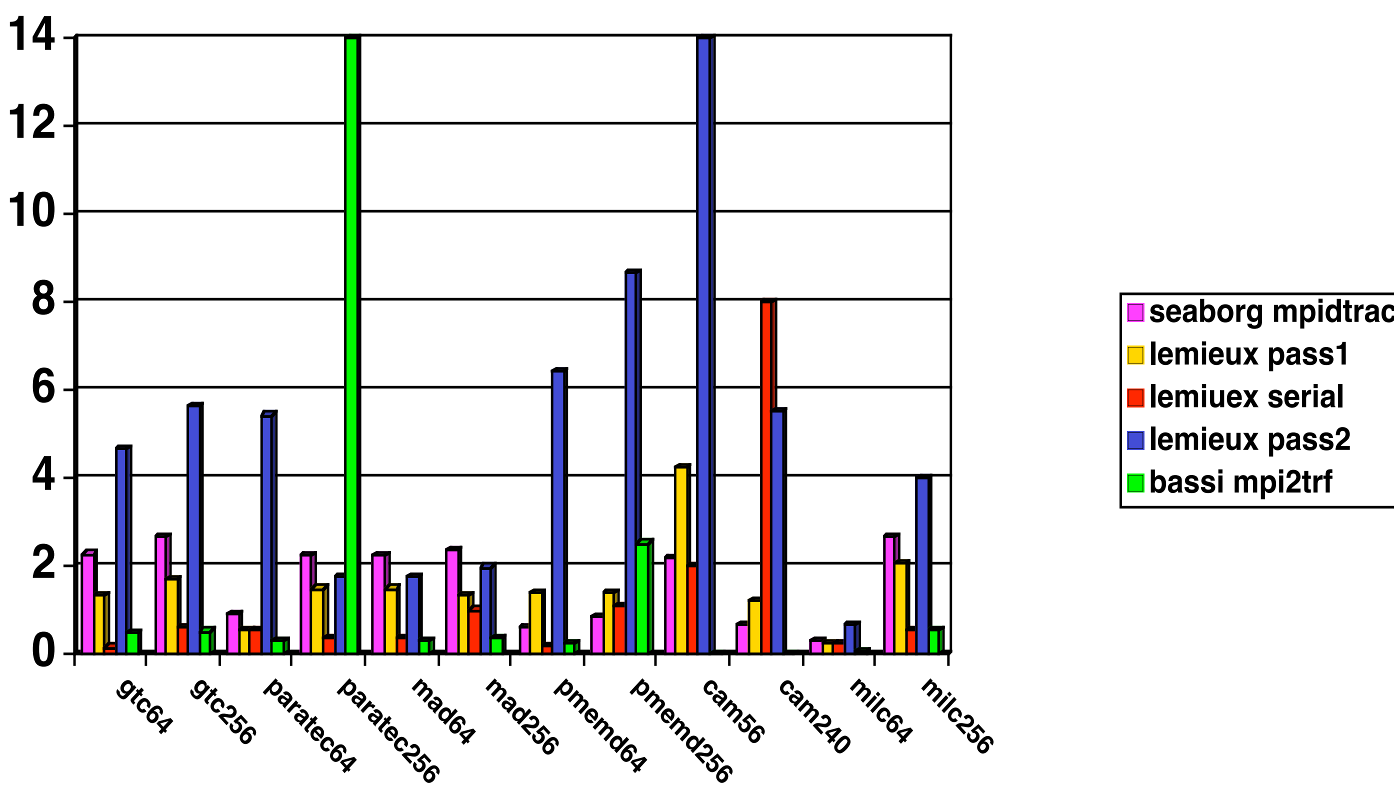

Figure 1. Runtimes of Various Steps (hours)

The file storage needs for this work are 1 GB total for all MetaSim tracing and about 300 GB for all mpidtraces that were successfully performed. Recall the cam application could not be run using mpidtrace. Also, the large (256-way) paratec benchmark was not traced with mpidtrace as the predicted file storage required was 3 TB and the predicted runtime was about 24 hours. The decision was made to continue prediction work with all other application-size pairs and, if successful, return to the large paratec run at a later date. The mpi2trf step, which combines data files into on large trace file is a serial process and can take more than 12 hours, even on bassi, which is an IBM POWER5 machine.

Obtain memory and network speeds of target machines

Before beginning the convolving step, a few basic hardware measurements are obtained using small benchmark kernels on each target machine. MAPS (Memory Access Pattern Signature) is used to measure memory bandwidth (for L1, L2, and main memory) for several buffer data sizes and at stride one and random stride access patterns. Netbench is used to measure the latency and network bandwidth of a few CPU's of the target machine. The values of interest are on-node and off-node bandwidth and latency measurements. As described above, Dimemas uses only one value for each, for example the values for Seaborg are off-node: 317 MB/s (max), 29 μs (typical) and on-node: 406

MB/s (max), 10 μs (typical). MAPS and netbench must be run on each target machine under consideration.

For each machine, the MAPS and netbench measurements only need to be collected once. Most of this step is completed as PMaC already has a database of machine data. Various versions of the MAPS kernel were run on NERSC machines and results were reported to PMaC.

Summary

In summary, the Metasim tracer was used to collect data for six NERSC-5 applications on Lemieux. Mpidtrace was used to collect communication data for five NERSC-5 applications on Seaborg. Hardware data was obtained using MAPS and netbench for NERSC machines. A directory was created on the NERSC global filesytem (NGF) to store the data and be accessible to the PMaC team. We are currently waiting for analysis by PMaC.

No prediction results have been made for the N5 applications on any target machines. As far as I know, the progress on this effort is the same as it was last summer (2006) and the remaining work was to be completed by the PMAC group at SDSC. My primary interaction was with Allan Snavely and Laura Carrington at SDSC.

Some of the complications experienced while using the framework were:

1) Difficult to obtain accurate cache system data for new machines
2) Also newer machines can be difficult to access, which is required to runs MAPS and netbench.
3) All applications must be ported and run on Alpha processor machine
4) All applications must be run at full scale multiple times (at least twice for memory tracing and once for communication tracing)
5) Requires commercial network package, which did not always work properly (failed with the cam benchmark and had other difficulties)
6) The method requires multiple steps and there is no way to verify that each step was completed properly
7) Some parameters to be used as input to the convolving step were not easily determined.
8) Resources required to collect and process tracing data was significant

Conclusions

The PMAC framework is intended to predict system performance without actually running the benchmark codes on the target systems by using memory & communication trace data collected from the code combined together with a high-level memory trace and interconnect simulation of the target machine architectures. The goal of the project was to see if the PMAC framework was capable of correctly predicting the performance of the

NERSC SSP benchmarks on existing machines and also machines that had been bid for NERSC-5The benchmark data for the NERSC-5 procurement was not shared with us so as not to bias the study and to comply with NDA requirements.

It is the conclusion of this study that the amount of work required to use the PMAC framework far exceeds that required to simply run the application benchmarks on the target architectures. This is true both in terms of human resources to collect the data and run various components of the framework and in terms of total computational resources required to run the framework. It does not reduce the effort required by vendors or by the procurement team to compare systems as far as we were able to determine.

The framework is not capable of predicting the performance of systems that are larger than those used to collect the data. That is to say, you cannot use data collected from a 128-way example run on a smaller system to predict performance for much larger concurrencies. This is also one of the key impediments to system assessment during procurements, which is not solved by the PMAC prediction model.

Finally, the framework does not account for differences in compiler performance on different systems. For example, the code generated by the pathscale compiler on the 2.2Ghz opteron processors of jaquard can be up to 30% faster than code generated by the PGI compiler for the much faster 2.6Ghz processors of the jaguar system at ORNL. The framework only does memory traces of the code as compiled on a single system, without regard to the differences in prefetch strategies emitted by different proposed compilers. This can also result in significant performance differences between prediction and reality that are not treated by the framework.

PMAC has since abandoned this framework and has moved to a new framework for performance prediction (http://www.sdsc.edu/pmac/projects/index.html). The differences between this and the older framework are significant enough to render our old work obsolete at this point in time. Further progress on this evaluation would entail dropping all of the currently collected data and starting over again with the new framework.

Acknowledgements
We would like to acknowledge the assistance of Alan Snavely and Laura Carrington of the PMaC group at SDSC for all of their assistance in configuring the framework and collecting data. The work was performed under DOE/LBNL contract number DE-AC02-05CH11231.

Author Affiliation
The author works for the Applied Numerical Algorithms Group (ANAG) in the Computing Research Division of Lawrence Berkeley National Laboratory. The work was directed and funded by the NERSC Center Division of LBNL as part of the NERSC-5 Procurement.